A text editor is a program for creating and modifying text documents. A line editor is a very simple type of editor. The file EditorApplication.java contains a line editor program that supports the following operations: **INSERT** a new line of text into the document; **DELETE** the current text line from the document; Move UP one line in the document; Move DOWN one line in the document; Display the **CURRENT** line of text; Display ALL the lines of the entire document; **QUIT** editing the document.

The file EditorApplication.java includes only the user interface of the line editor. The document itself is constructed and modified by methods of the Editor class. The methods of the Editor class are not implemented. You must implement them, according to the instructions below. Your implementation of the Editor methods will make use of several existing methods of the ListDL class, as well as some new ones. You must implement the new methods of the ListDL class, according to the instructions below.

I. Methods of the Editor class:

This class maintains the document as a doubly linked list (ListDL object), stored in an instance variable called “text”. It keeps track of the current line of text using an integer instance variable. The current line ranges from 0 up to text.size(). It may help to think of this variable as referring to separators lying in between the lines of the document, as indicated by the diagram below:

```
0      1      2      3      ...      N-1      N
```

The CURRENT, DELETE and INSERT operations all make use of the current line. CURRENT displays the current line. DELETE removes the current line. For these two operations to be carried out, current must lie in the range 0 ... text.size()-1. INSERT adds a new line of text at the position indicated by the arrow, i.e., just before the current line. For this operation to be carried out, current must lie in the range 0 ... text.size(). The Editor class also has an IOConsole instance variable console that is used for printing text.

- **Editor(IOConsole console)** Method to construct an editor containing no text: Initialize the console variable to be the parameter to the constructor. Initialize text to be an empty list, using the default constructor for the ListDL class. Initialize current to be zero. Initialize the console variable to be the parameter to the constructor. Note: The console variable represents the window to which print statements should send their output, e.g., using statements such as console.print(“Message”); or console.println(“Message”);.
• public void up() Method to change the current line of text to be the one above the current line: If current is greater than zero, then decrement current, otherwise print out a message saying that the editor cannot move up any more.

• public void down() Method to change the current line of text to be the one below the current line: If current is less than the number text.size() of lines in the document, then increment current, otherwise print out a message saying that the editor cannot move down any more.

• public void insert(String newLine) Method to insert line into the document, just before the current line. Use the insert method of the ListDL class. Set the new current line to be the old current line plus one.

• public void delete() Method to delete the current line: If current < text.size(), then use the delete method of the ListDL class to remove the current line from the document. Otherwise, print out a message saying there is no line to delete.

• public void printCurrent() Method to print out the current line: If current < text.size(), then use the elementAt method of the ListDL class to get current line, and print it out. Otherwise, print out a message saying there is no line to print.

• public void printAll() Method to print out the entire document: Use the elementAt method of the ListDL class to access and print each line of the document. Print each line of the document on one line of the screen. Print “ “ (two spaces) at the beginning of each line, except for the current line. Print “> ” at the beginning of the current line. If the current is text.size(), i.e., the current line is one past the last line of the document, then print “> ” after printing the last line of the document.

II. Methods of the ListDL class:

• public Object elementAt(int index) Method to return the value stored in the ListElementDL object at position index of this list: Positions run from 0 (head of the list) up to count-1 (tail of the list). Assume that index lies in this range. Start at the head of the list. Repeatedly use the next method of the ListElementDL class to move an appropriate distance down the list. Use the data method of the ListElementDL class to get the data stored in the appropriate position.

• public void delete(int index) Method to remove the ListElementDL object at position index of this list: Positions run from 0 (head of the list) up to count-1 (tail of the list). Assume that index lies in this range. First find the ListElementDL object at position index. Use the setPrevious method of the ListElementDL class to arrange that the previous field of the element at position index+1 points back to the element at position index-1. Use the setNext method of the ListElementDL class to arrange that the next field of the element at position index-1 points forward to the element at position index+1. Remember to handle the special cases in which index refers to the head of the list, or to the tail of the list, or both. (Our assumption that index is a valid position implies that this list is initially not
empty, so you don’t need to handle that case.) Finally, decrement count, indicating that the list has one fewer elements.

- **public void insert(Object data, int index)** Method to insert data into this list object, just prior to the index position, i.e., in between positions index-1 and index: Positions run from 0 (head of the list) up to count (one position past the tail of the list). Assume that index lies in this range. First make a new ListElementDL object to hold data and save it in a local variable: newElement. Then find the ListElementDL object at position index and save it in a variable: current. Also let prior be the ListElementDL object that is just prior to current. Then do the following:
  1. Set the previous field of current to be newElement.
  2. Set the next field of newElement to be current.
  3. Set next field of prior to be newElement.
  4. Set the previous field of newElement to be prior.
Remember to handle the special cases in which index refers to the head of the list, or to the position after the tail of the list, or both. Also remember to handle the special cases in which this list is empty. Finally, add one to count, indicating that the size of the list has changed.

**III.** Going Generic:

- Extend the List interface to include the insert, delete and elementAt operations you implemented above.

- Modify List interface, the ListElementDL class and the ListDL class so they each use a parameterized type D instead of Object for the type of data stored in the list.

- Modify the Editor class to use the parameterized List interface and parameterized ListDL class.
EditorApplication Execution Trace

Line Editor Program
Command: insert
Insert: Now is the time for all
Command: insert
Insert: good men to come to the
Command: all
Now is the time for all
good men to come to the
>
Command: up
Command: up
Command: current
Now is the time for all
Command: delete
Command: insert
Insert: Now is the time for all
Command: all
Now is the time for all
> good men to come to the
Command: down
Command: all
Now is the time for all
good men to come to the
>
Command: insert
Insert: aid of their country.
Command: all
Now is the time for all
good men to come to the
aid of their country.
>
Command: up
Command: up
Command: delete
Command: insert
Insert: good people to come to the
Command: all
Now is the time for all
good people to come to the
> aid of their country.
Command: